paralist[*]
Extended List Environments

Bernd Schandl
schandl@gmx.net

Printed on May 9, 2005

Abstract

This package provides some new list environments. Itemized and enumerated lists can be typeset within paragraphs, as paragraphs and in a compact version. Most environments have optional arguments to format the labels. Additionally, the LaTeX environments `itemize` and `enumerate` can be extended to use a similar optional argument.

1 Introduction

In a posting to `comp.text.tex` in May 1998, someone asked about the possibility of an enumerated environment that (a) can be used within paragraphs, (b) takes care of enumeration and (c) has items that can be referenced. Another posting mentioned the package theapa as a possible solution. Now that I was looking for that kind of environment and found those old postings, I had a look at theapa and decided to take out the part about list environments and rewrite it a little bit.

Over time, compact versions of `enumerate`, `itemize` and `description` have been added and optional arguments for most environments make it possible to define a special way of formatting the labels.

2 Package Options

Certain parts of the package are only executed if appropriate options are specified.

newitem/olditem With newitem (set by default), the LaTeX environment `itemize` will be redefined to have an optional argument to specify the format of the label. See Section 3. Specifying olditem will leave `itemize` as it is.

[*]Package version v2.3b of 2002/03/18.

newenum/oldenum With newenum (set by default), the LaTeX environment `enumerate` will be redefined to have an optional argument to specify the format of the label. See Section 3. Specifying oldenum will leave `enumerate` as it is.

alwaysadjust The width of the labels of the environments `compactenum`, `enumerate`, `compactitem` and `itemize` is always adjusted to the actual label. For the default labels, this means that the label width is usually decreased.

neveradjust The width of the labels is never adjusted, not even for environments where you defined the labels manually using the optional argument. This option is ignored if option alwaysadjust is used as well.

neverdecrease If the width of the labels is adjusted, this option avoids the decrease of the width. Here is an example why this might make sense. If no ...adjust option is specified, then the indentation of the `\item` in `\begin{enumerate}` and `\begin{enumerate}[1.]` is different although they have the same labels.

defblank The two environments `inparablank` and `asparablank` will be defined. See Section 5.4.

pointlessenum The items in the enumerated environments are labeled and referenced as in "1", "1.1", "1.1.1" and "1.1.1.1". See also Section 3.

pointedenum The items in the enumerated environments are labeled as in "1.", "1.1.", "1.1.1." and "1.1.1.1." and referenced without the trailing point. See also Section 3. This option is ignored if pointlessenum is used.

flushright The labels in the four lists mentioned above are set flush right. As this is the LaTeX default, this is also the default for this package.

flushleft The labels in the four lists mentioned above are set flush left.

cfg The configuration file `paralist.cfg` is loaded if it exists. (default)

nocfg The configuration file is not loaded.

The option increaseonly is deprecated; use neverdecrease instead.

3 Formatting the Labels

All the itemized and enumerated environments have optional arguments to specify the format of the labels. The following examples will only work if you have loaded paralist *without* the options olditem and oldenum.

Using the LaTeX standard classes, `itemize` uses the following symbols for the labels of the four list levels: $\bullet$ $-$ $\ast$ $\cdot$. If you want to change this for a particular environment, just say something like

```
\begin{itemize}[$\star$]
```

and $\star$ will be used instead of the default symbol.

The optional argument of the enumerated environment is a little more complicated, but whoever has worked with David Carlisle's enumerate package can skip the rest of the section since exactly the same mechanism (and almost the same code) is used.

The tokens `A`, `a`, `I`, `i`, and `1` can be used in the optional argument to produce the counter with one of the styles `\Alph`, `\alph`, `\Roman`, `\roman` and `\arabic`.[1] These letters may be surrounded by any other string involving any other TeX expression. However, the tokens `A a I i 1` must be inside a `{ }` group if they should not be taken as special. A few examples follow.

```
\begin{enumerate}[(i)]
```
produces the labels (i), (ii), (iii) ...

```
\begin{enumerate}[{example} a)]
```
produces example a), example b), example c) ...

```
\begin{enumerate}[{A}-1]
```
produces A-1, A-2, A-3 ...

```
\begin{enumerate}[\bfseries {Item} I]
```
produces **Item I**, **Item II**, **Item III** ...

Note that in the last example `[\textbf{Item I}]` does not work because the special token `I` is inside a group.

The `\ref` command produces only the counter without the surrounding text, so in the examples above you would get i, a, 1 and I respectively if you referenced the first item.

`\pointedenum`
`\pointlessenum`

There are also two package options and two corresponding macros to format the labels and references of enumerated environments. The option pointedenum and the macro `\pointedenum` format the labels as in "1.", "1.1.", "1.1.1." and "1.1.1.1." and the references without the trailing point. The option pointlessenum and the macro `\pointlessenum` do not use the trailing point in the labels either.

While the package options make a global change, the macros can be used for a local change or to define a special environment, for example

```
\newenvironment{myenum}%
  {\pointedenum\begin{enumerate}}%
  {\end{enumerate}}
```
Note that `\begin{enumerate}` is used *after* `\pointedenum`, otherwise the optional argument of enumerate would not work (in case you want to use them within the same environment which doesn't really make sense).

`\paradescriptionlabel`

In the document classes, the label format for the `description` environment is defined as

```
\newcommand*\descriptionlabel[1]{%
  \hspace\labelsep\normalfont\bfseries #1}.
```
This is also used by `compactdesc`. For the environments `inparadesc` and `asparadesc`, there is a separate macro called `\paradescriptionlabel` de-

[1]The set of tokens can be extended. Look for `\pl@hook` in the code section.

fined (almost) like this:

```
\newcommand*\paradescriptionlabel[1]{%
  \normalfont\bfseries #1}.
```

4 Defaults for Labels and Margins

If you want your lists labeled differently than the LaTeX default throughout your document, it is a bit awkward to use the optional argument of the environments all the time. Therefore three macros are provided to define the labels and the left margins of the list environments.

Note that the macros defining the labels do *not* adapt the left margins of the list environments because this may have unexpected side effects. If you want that change, you have to explicitly use `\setdefaultleftmargin`.

If in any of the following three macros an argument is empty, then the according label or margin is left unchanged.

`\setdefaultitem` The default labels of itemized environments can be set by using the macro `\setdefaultitem` which needs four arguments. To get the LaTeX default labels say

```
\setdefaultitem{\textbullet}%
  {\normalfont\bfseries \textendash}%
  {\textasteriskcentered}{\textperiodcentered}
```

(which is of course silly because you don't need to do anything if you want to stick with the default labels). If you want a triangle ($\triangleright$) instead of the endash for level two just say

```
\setdefaultitem{}{$\triangleright$}{}{}.
```

`\setdefaultenum` The labels of enumerated lists are formatted with `\setdefaultenum` using the mechanism described in Section 3. The LaTeX default labels could be defined by

```
\setdefaultenum{1.}{(a)}{i.}{A.}.
```

If you want capital Roman letters for level three, just say

```
\setdefaultenum{}{}{I.}{}.
```

`\setdefaultleftmargin` To change the left margin of the lists, use `\setdefaultleftmargin`. The length `\leftmargin`n specifies the indentation of a list of level n with respect to the list of level $n-1$ or the surrounding text (if $n = 1$). The environments that use `\leftmargin`n are (at least) `enumerate`, `compactenum`, `itemize` and `compactitem` (maybe a few more that I am not aware of). The LaTeX settings could be defined by

```
\setdefaultleftmargin{2.5em}{2.2em}{1.87em}{1.7em}{1em}{1em}.
```

In twocolumn mode LaTeX uses a smaller margin for the first, fifth and sixth level which could be defined by

```
\setdefaultleftmargin{2em}{}{}{}{.5em}{.5em}.
```

The macros `\defaultitem`, `\defaultenum` and `\defaultleftmargin` should not be used anymore. They are only kept for backward compatibility with package versions < 2.1.

If some of your changes should appear in *every* document that uses paralist, put them in a file `paralist.cfg` which is read at the end of the package in case it exists (unless you specified the option nocfg).

5 New Environments

5.1 Enumerated Environments

`asparaenum` The environment `asparaenum` is an enumerated environment in which the items are formatted as separate paragraphs.

As an example, we use `asparaenum` within this paragraph.

1. Every `\item` is basically set as a separate paragraph. The second line is *not* indented (this is a feature, not a bug).

2. The next `\item` looks like this and is labeled.

The example was produced by the following piece of code:

```
\begin{asparaenum}
  \item Every ...
  \item The next ... \label{pl1}
\end{asparaenum}
```

By saying `\ref{pl1}` we get 2.

`inparaenum` The `inparaenum` environment formats an enumerated list within a paragraph, just like the one in the introduction.

The example in the introduction was set by the following commands:

```
... of an enumerated environment that
\begin{inparaenum}[(a)]
  \item can be used within paragraphs,
  \item takes care of enumeration and
  \item has items that can be referenced. \label{pl2}
\end{inparaenum}
Another posting mentioned ...
```

By saying `\ref{pl2}` we get c.

`compactenum` The `compactenum` environment is just a compact version of the standard `enumerate` environment. All the vertical skips are set to zero (actually they are adjustable, see Section 7).

5.2 Itemized Environments

`asparaitem` The `asparaitem` environment is very similar to `asparaenum`. It just uses symbols instead of enumerating the items. The environment has an optional argument which specifies the symbol. For an example see Section 6.

`inparaitem` Similar to `inparaenum` I added an environment `inparaitem` which also has an optional argument. I don't really know why anybody would use it, but I added it because of symmetry.

compactitem The compactitem environment is again just a compact version of the standard itemize environment with all the vertical skips set to zero. So by using this environment
- ◦ you can save some space and
- ◦ specify the symbol.
- ◦ Let me add a longer item so that you can see that we have a different indentation than in the asparaitem environment.

The code of the example above is

```
\begin{compactitem}[$\circ$]
  \item you can save some space and
  \item specify the symbol.
  \item Let me add ...
\end{compactitem}
```

5.3 Descriptive Environments

asparadesc The asparadesc environment is again very similar to asparaenum. It just uses the optional arument of \item as the "intro" for the paragraph.

inparaitem Again similar to inparaenum, I added an environment inparadesc. Probably nobody would use it but I added it because of symmetry.

compactdesc The compactdesc environment is copied from the LaTeX standard classes with all the vertical skips set to zero. By the way, does anybody know why description has to be defined by the document class and is not defined in ltlists.dtx?

5.4 Blank Environments

Someone requested list environments that print their items as if there was no list. It seems that this makes entering structured data a little easier in LyX. Since not everybody needs these (odd) environments they are only defined if the package is loaded with the option defblank. The following two environments do not have optional arguments because there is no label to format.

asparablank Every item is formatted just as if it was a regular paragraph. If you want to use the optional argument of \item, you have to add some white space at the end because \labelsep is set to zero. Use something like

\item[\textbullet\hspace{.5em}]

inparablank The items are set just as regular text. The "white space problem" mentioned in the last paragraph is handled automatically. If I didn't tell you, you wouldn't know that this paragraph is set using the following construction:

```
... are set
```

```
\begin{inparablank}
  \item just as ...
  \item The ...
  \item is handled ...
\end{inparablank}
If I didn't ...
```

6 Nesting Environments

All the environments can be nested just as the standard list environments although the results might sometimes not be as expected. For example, it's probably not a good idea to call another list environment within a `inpara...` environment, but why should anyone want to do this? The maximal nesting level is six (four of the same kind), just as for the LaTeX environments.

This paragraph is

$\star$ an example for the usage of `asparaitem` and its optional argument,

$\star$ and a demonstration that (i) you can use `inparaenum` within `asparaitem` and (ii) you can still reference it.

The reference was in subitem (ii). The code of the last example is

```
\begin{asparaitem}[$\star$]
  \item an example ...
  \item and a demonstration that
    \begin{inparaenum}[(i)]
      \item you can use ...
      \item can still ... \label{pl3}
    \end{inparaenum}
\end{asparaitem}
The reference was in subitem (\ref{pl3}).
```

7 Fine-Tuning

Ok, I already hear someone saying "Your compact lists are a nice idea, but I'd like to have it a little less compact." Here is a solution. The following skips can be adjusted using `\setlength` and affect the spacing of the `compact...` environments. The names are chosen similar to the LaTeX names, so I just copy the explanation from `ltlists.dtx`.

`\pltopsep`: Space between first item and preceding paragraph.

`\plpartopsep`: Extra space added to `\topsep` when environment starts a new paragraph (is called in vmode).

`\plitemsep`: Space between successive items.

`\plparsep`: Space between paragraphs within an item – the `\parskip` for this environment.

Actually, the two `...topsep` skips are added before *and after* the list.

The default value for all of them is 0 pt. It is probably a good idea to define them depending on the font size if they are non-zero, i. e. using units such as `ex` or `em`.

8 Bugs and Wishes

No bugs . . . that I know of.

Well, there is actually one issue if you use the babel package with one of the options acadian, canadien, francais, frenchb or french (which all do essentially the same). Since it redefines the itemize environment at the `\start{document}`, the definition of `itemize` made by paralist is lost. There are three possible fixes:

1. Accept the `itemize` environment without optional argument. `:(`
2. Use `\FrenchItemizeSpacingfalse` after loading babel which will avoid the redefinition of `itemize` by babel. `:/`
3. Figure out a way to combine the code in babel and paralist and send the solution to me. `:)`

Feel free to let me know about any problems, suggestions and wishes you have concerning this package and its documentation. Praise is welcome, too `;-)` The most recent version of this package can always be found on CTAN or at `http://schandl.gmxhome.de/paralist/`.

9 Acknowledgments

I want to thank all the users who helped me with their comments finding bugs and extending the package. A big "Thank you" goes to David Carlisle, because there wouldn't be any optional arguments for the enumerated environments without the code from his enumerate package. Some pieces of code of the `inpara...` environments are inspired by Mogens Lemvig Hansen's shortlst package.

10 Implementation

At the beginning of this file, the `\ProvidesPackage` macro was executed. So we only need to state that we need LaTeX 2ε.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
```

10.1 Package Options

First we define the package options. The option increaseonly is only kept for compatibility.

```
\newif\if@plnewitem\@plnewitemtrue
\newif\if@plnewenum\@plnewenumtrue
\newif\if@plalwaysadjust\@plalwaysadjustfalse
\newif\if@plneveradjust\@plneveradjustfalse
\newif\if@plneverdecrease\@plneverdecreasefalse
\newif\if@pldefblank\@pldefblankfalse
\newif\if@plpointedenum\@plpointedenumfalse
\newif\if@plpointlessenum\@plpointlessenumfalse
\newif\if@plflushright\@plflushrighttrue
\newif\if@plloadcfg\@plloadcfgtrue
\DeclareOption{newitem}{\@plnewitemtrue}
\DeclareOption{olditem}{\@plnewitemfalse}
\DeclareOption{newenum}{\@plnewenumtrue}
\DeclareOption{oldenum}{\@plnewenumfalse}
\DeclareOption{alwaysadjust}{\@plalwaysadjusttrue}
\DeclareOption{neveradjust}{\@plneveradjusttrue}
\DeclareOption{neverdecrease}{\@plneverdecreasetrue}
\DeclareOption{increaseonly}{\@plneverdecreasetrue
  \PackageWarningNoLine{Paralist}{Option increaseonly deprecated.
    \MessageBreak Use option neverdecrease instead}}
\DeclareOption{defblank}{\@pldefblanktrue}
\DeclareOption{pointedenum}{\@plpointedenumtrue}
\DeclareOption{pointlessenum}{\@plpointlessenumtrue}
\DeclareOption{cfg}{\@plloadcfgtrue}
\DeclareOption{nocfg}{\@plloadcfgfalse}
\DeclareOption{flushright}{\@plflushrighttrue}
\DeclareOption{flushleft}{\@plflushrightfalse}
\ExecuteOptions{newitem,newenum,cfg,flushright}
\ProcessOptions\relax
```

Make sure that alwaysadjust overwrites neveradjust.

```
\if@plalwaysadjust\@plneveradjustfalse\fi
```

10.2 New Skips

The new skips are all set to zero.

```
\newlength{\pltopsep}
\newlength{\plpartopsep}
\newlength{\plitemsep}
\newlength{\plparsep}
\setlength{\pltopsep}{0pt}
\setlength{\plpartopsep}{0pt}
\setlength{\plitemsep}{0pt}
\setlength{\plparsep}{0pt}
```

10.3 Little "Helper" Macros

\if@empty A small macro to detect whether #1 is empty.

```
\def\if@empty#1#2#3{%
```

```
  \def\@tempa{#1}%
  \ifx\@tempa\@empty#2\else#3\fi}
```

`\pl@item` The following macro is used for all `inpara...` environments. It adds space after the label only when the label has a positive width.

```
\def\pl@item[#1]{%
  \if@noitemarg
    \@noitemargfalse
    \if@nmbrlist
      \refstepcounter{\@listctr}%
    \fi
  \fi
  \settowidth{\@tempdima}{#1}%
  \ifdim\@tempdima>\z@\makelabel{{#1}}\nobreakspace\fi
  \ignorespaces
  }
```

`\paradescriptionlabel` Similar to the LaTeX macro `\descriptionlabel` we define a macro for the description environments in and as paragraphs.

```
\def\paradescriptionlabel#1{{\normalfont\bfseries #1}}
```

`\enumlabel` Similar to the LaTeX macro `\descriptionlabel` we define a macro for the enumerated environments depending on whether the label should be set flush left or flush right.

```
\if@plflushright
  \def\enumlabel#1{\hss#1}
\else
  \def\enumlabel#1{#1\hfil}
\fi
```

`\itemlabel` Similar to the LaTeX macro `\descriptionlabel` we define a macro for the itemized environments depending on whether the label should be set flush left or flush right.

```
\if@plflushright
  \def\itemlabel#1{\hss#1}
\else
  \def\itemlabel#1{#1\hfil}
\fi
```

10.4 Adjusting the Label Width

The width of the label is only adjusted either if the format of the label was specified manually and the option neveradjust was *not* used or if the option alwaysadjust was used. Then `\leftmargin`n is set to the width of the label plus `\labelsep`. The width of the label is only decreased if the option neverdecrease was *not* used.

First we need some flags to indicate whether the label was redefined by the environment and whether the label should be adjusted.

```
\newif\if@plmylabel\@plmylabelfalse
\newif\if@pldoadjust\@pldoadjustfalse
```

`\@adjust@enum@labelwidth` For the enumerated environments, the counter is first set to 7 so that we get the width of 'vii' if Roman numbering is in force.

```
\def\@adjust@enum@labelwidth{%
  \if@plneveradjust\else
    \if@plalwaysadjust\@pldoadjusttrue\else
      \if@plmylabel\@pldoadjusttrue\fi
    \fi
  \fi
  \if@pldoadjust
    \csname c@\@enumctr\endcsname7
    \settowidth{\@tempdima}{%
      \csname label\@enumctr\endcsname\hspace{\labelsep}}%
```

The `\leftmargin`n has to be adjusted (where n is `\@listdepth`) but we first have to increase `\@listdepth` by one since the list has not yet started. This is reset right afterwards.

```
    \advance\@listdepth by 1\relax
    \if@plneverdecrease
      \ifdim\@tempdima >
        \csname leftmargin\romannumeral\@listdepth\endcsname
          \csname leftmargin\romannumeral\@listdepth\endcsname
            \@tempdima
      \fi
    \else
      \csname leftmargin\romannumeral\@listdepth\endcsname
        \@tempdima
    \fi
    \advance\@listdepth by -1\relax
  \fi
  \@pldoadjustfalse
  \@plmylabelfalse}
```

`\@adjust@item@labelwidth` For itemized labels, no counter needs to be set. The label is hidden in a macro the name of which is stored in `\@itemitem`.

```
\def\@adjust@item@labelwidth{%
  \if@plneveradjust\else
    \if@plalwaysadjust\@pldoadjusttrue\else
      \if@plmylabel\@pldoadjusttrue\fi
    \fi
  \fi
  \if@pldoadjust
    \settowidth{\@tempdima}{%
      \csname\@itemitem\endcsname\hspace{\labelsep}}%
```

The `\leftmargin`*n* has to be adjusted (where *n* is `\@listdepth`) but we first have to increase `\@listdepth` by one since the list has not yet started. This is reset right afterwards.

```
    \advance\@listdepth by 1\relax
    \if@plneverdecrease
      \ifdim\@tempdima >
        \csname leftmargin\romannumeral\@listdepth\endcsname
          \csname leftmargin\romannumeral\@listdepth\endcsname
            \@tempdima
      \fi
    \else
      \csname leftmargin\romannumeral\@listdepth\endcsname
        \@tempdima
    \fi
    \advance\@listdepth by -1\relax
  \fi
  \@pldoadjustfalse
  \@plmylabelfalse}
```

10.5 Label of Enumerated Environments

It follows all the stuff for the optional argument of the enumerated environments. The code and most of its documentation is basically copied from David Carlisle's enumerate package.

`\pl@lab` First we define an internal token register which is used to build up the label command from the optional argument.

```
\newtoks\pl@lab
```

`\pl@qmark` This just expands to a '?'. `\ref` will produce this, if no counter is printed.

```
\def\pl@qmark{?}
```

The next four macros build up the command that will print the label. They each gobble one token or group from the optional argument and add corresponding tokens to the register `\pl@lab`. They each end with a call to `\pl@loop`, which starts the processing of the next token.

`\pl@label` Add the counter to the label. Argument `#2` will be one of the 'special' tokens `A a I i 1`, and is thrown away. `#1` will be a command like `\Roman`. In every call `\@tempcnta` is increased, so we can check at the end that exactly one counter was defined.

```
\def\pl@label#1#2{%
  \edef\pl@the{\noexpand#1{\@enumctr}}%
  \pl@lab\expandafter{\the\pl@lab\csname the\@enumctr\endcsname}%
  \advance\@tempcnta1
  \pl@loop}
```

\pl@space \pl@sp@ce Add a space to the label. The tricky bit is to gobble the space token, as you can not do this with a macro argument.

```
\def\pl@space{\afterassignment\pl@sp@ce\let\@tempa= }
\def\pl@sp@ce{\pl@lab\expandafter{\the\pl@lab\space}\pl@loop}
```

\pl@group Add a `{ }` group to the label.

```
\def\pl@group#1{\pl@lab\expandafter{\the\pl@lab{#1}}\pl@loop}
```

\pl@other Add anything else to the label.

```
\def\pl@other#1{\pl@lab\expandafter{\the\pl@lab#1}\pl@loop}
```

\pl@loop \pl@loop@ The body of the main loop. Eating tokens this way instead of using `\@tfor` lets you see spaces and *all* braces. `\@tfor` would treat `a` and `{a}` as special, but not `{{a}}`.

```
\def\pl@loop{\futurelet\pl@temp\pl@loop@}
\def\pl@loop@{%
  \ifx A\pl@temp         \def\@tempa{\pl@label\Alph  }\else
  \ifx a\pl@temp         \def\@tempa{\pl@label\alph  }\else
  \ifx i\pl@temp         \def\@tempa{\pl@label\roman }\else
  \ifx I\pl@temp         \def\@tempa{\pl@label\Roman }\else
  \ifx 1\pl@temp         \def\@tempa{\pl@label\arabic}\else
  \ifx \@sptoken\pl@temp \let\@tempa\pl@space         \else
  \ifx \bgroup\pl@temp   \let\@tempa\pl@group         \else
  \ifx \@@@\pl@temp      \let\@tempa\@gobble          \else
                         \let\@tempa\pl@other
```

Hook for possible extensions.

```
                         \pl@hook
                 \fi\fi\fi\fi\fi\fi\fi\fi
```

Process the current token, then look at the next.

```
  \@tempa}
```

\pl@hook Hook for possible extensions. Some packages may want to extend the number of special characters that are associated with counter representations. You can use e. g. Greek or Russian characters. Here is an example of a footnote symbol counter, triggered by `*`.

To enable a new counter type based on a letter, you just need to add a new `\ifx` clause similar to the code above. To make `*` trigger footnote symbol counting do the following.

Initialize the hook, in case the package is loaded before paralist.

```
\providecommand\pl@hook{}
```

Add to the hook a new `\ifx` clause that associates `*` with the `\fnsymbol` counter command.

```
\g@addto@macro\pl@hook{%
  \ifx *\pl@temp
    \def\@tempa{\pl@label\fnsymbol}%
  \fi}
```

This code sequence should work whether it is loaded before or after the paralist package. Any number of new counter types may be added in this way.

At this point we just need to initialize the hook, taking care not to overwrite any definitions another package may already have added.

```
\providecommand\pl@hook{}
```

`\@enumlabel@` The following macro handles the optional argument of an enumerated environment. Its first argument is the macro that should be called after `\@enumlabel@` has done the job, the second one is the optional argument from the calling environment.

```
\def\@enumlabel@#1[#2]{%
```

The flag is set that the environment redefines its label.

```
  \@plmylabeltrue
```

We initialize the number of counter-defining tokens to 0. Then `\pl@lab` and `\pl@the` are initialized. Later, `\pl@the` will be used to define `\theenum`n.

```
  \@tempcnta0
  \pl@lab{}%
  \let\pl@the\pl@qmark
```

Now we analyze the second argument using a loop. As we will explain in the definition of `asparaenum`, we had to add an extra bracket `[` to the second argument, which we have to gobble before we start processing it. The macro `\@@@` is never expanded (or even defined), it is just used to detect the end of the token list.

```
  \expandafter\pl@loop\@gobble#2\@@@
```

If there was no special token or several special tokens we issue a warning.

```
  \ifnum\@tempcnta=1\else
    \PackageWarning{paralist}{Incorrect label; no or multiple
      counters.\MessageBreak The label is: \@gobble#2}%
  \fi
```

Define `\labelenum`n and `\theenum`n.

```
  \expandafter\edef\csname label\@enumctr\endcsname{\the\pl@lab}%
  \expandafter\let\csname the\@enumctr\endcsname\pl@the
```

Finally call the macro that finishes the `\begin` part of the calling environment.

```
  #1}
```

10.6 Label of the Itemized Environment

`\@itemlabel@` The handling of the optional argument of itemized environments is a lot easier.

```
\def\@itemlabel@#1[#2]{%
  \@plmylabeltrue
  \def\pl@itemitem{#2}%
  \def\@itemitem{pl@itemitem}%
  #1}
```

10.7 Enumerated Environments

asparaenum

The first environment formats the enumerated items as paragraphs. Most of the skips are set to zero, the items are indented as if they were paragraphs and `\makelabel` is redefined so that it just prints the counter.

```
\def\asparaenum{%
  \ifnum\@enumdepth>\thr@@
    \@toodeep
  \else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
  \fi
```

In the following line the second bracket `[` seems to be superfluous. Here is why we need it. If TEX finds an argument in curly braces then it strips of the braces before handing the argument to the macro `\@enumlabel@`. So the optional argument `[{\bfseries a}]` would be accepted, although the counter is in a group. By adding the bracket to the argument we will never have braces around the whole argument. In the macro `\@enumlabel@` the bracket is gobbled before the processing of the argument starts.

```
  \@ifnextchar[{\@enumlabel@{\@asparaenum@}[}{\@asparaenum@}}
\def\@asparaenum@{%
  \expandafter\list\csname label\@enumctr\endcsname{%
    \usecounter{\@enumctr}%
    \labelwidth\z@
    \labelsep.5em
    \leftmargin\z@
    \parsep\parskip
    \itemsep\z@
    \topsep\z@
    \partopsep\parskip
    \itemindent\parindent
    \advance\itemindent\labelsep
    \def\makelabel##1{##1}}}
\let\endasparaenum\endlist
```

inparaenum

Now the enumerated environment within a paragraph. Since it is not really a list we have to redefine `\@item`.

```
\def\inparaenum{%
  \ifnum\@enumdepth>\thr@@
    \@toodeep
  \else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
  \fi
  \@ifnextchar[{\@enumlabel@{\@inparaenum@}[]}{\@inparaenum@}}
\def\@inparaenum@{%
  \usecounter{\@enumctr}%
  \def\@itemlabel{\csname label\@enumctr\endcsname}%
  \let\@item\pl@item
  \def\makelabel##1{##1}%
  \ignorespaces}
\let\endinparaenum\ignorespacesafterend
```

compactenum For the `compactenum` environment all vertical skips have to be set to the `\pl...` values.

```
\def\compactenum{%
  \ifnum\@enumdepth>\thr@@
    \@toodeep
  \else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
  \fi
  \@ifnextchar[{\@enumlabel@{\@compactenum@}[]}{\@compactenum@}}
```

The width of the label is adjusted before starting the list if necessary.

```
\def\@compactenum@{%
  \@adjust@enum@labelwidth
  \expandafter\list\csname label\@enumctr\endcsname{%
    \usecounter{\@enumctr}%
    \parsep\plparsep
    \itemsep\plitemsep
    \topsep\pltopsep
    \partopsep\plpartopsep
    \labelwidth
      \csname leftmargin\romannumeral\@listdepth\endcsname
    \advance\labelwidth-\labelsep
    \let\makelabel\enumlabel}}
\let\endcompactenum\endlist
```

enumerate If the user has requested it, we redefine the `enumerate` environment.

```
\if@plnewenum
  \def\enumerate{%
    \ifnum \@enumdepth >\thr@@
      \@toodeep
    \else
      \advance\@enumdepth \@ne
      \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
    \fi
    \@ifnextchar[{\@enumlabel@{\@enumerate@}[]}{\@enumerate@}}
```

The width of the label is adjusted before starting the list if necessary.

```
  \def\@enumerate@{%
    \@adjust@enum@labelwidth
    \expandafter\list\csname label\@enumctr\endcsname{%
      \usecounter{\@enumctr}%
      \labelwidth
        \csname leftmargin\romannumeral\@listdepth\endcsname
      \advance\labelwidth-\labelsep
      \let\makelabel\enumlabel}}
\fi
```

10.8 Itemized Environments

asparaitem This is pretty much the same as asparaenum. We just don't need a counter.

```
\def\asparaitem{%
  \ifnum\@itemdepth>\thr@@
    \@toodeep
  \else
    \advance\@itemdepth\@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
  \fi
  \@ifnextchar[{\@itemlabel@{\@asparaitem@}}{\@asparaitem@}}
\def\@asparaitem@{%
  \expandafter\list\csname\@itemitem\endcsname{%
    \labelwidth\z@
    \labelsep.5em
    \leftmargin\z@
    \parsep\parskip
    \itemsep\z@
    \topsep\z@
    \partopsep\parskip
    \itemindent\parindent
    \advance\itemindent\labelsep
    \def\makelabel##1{##1}}}
\let\endasparaitem\endlist
```

inparaitem Again the same as inparaenum without the counter stuff but with an optional argument.

```
\def\inparaitem{%
  \ifnum\@itemdepth>\thr@@
    \@toodeep
  \else
    \advance\@itemdepth\@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
  \fi
  \@ifnextchar[{\@itemlabel@{\@inparaitem@}}{\@inparaitem@}}
\def\@inparaitem@{%
  \def\@itemlabel{\csname\@itemitem\endcsname}%
  \let\@item\pl@item
  \def\makelabel##1{##1}%
  \ignorespaces}
\let\endinparaitem\ignorespacesafterend
```

`compactitem` The `compactitem` environment is basically the same as the LaTeX original. All the vertical skips are set to the `\pl...` values.

```
\def\compactitem{%
  \ifnum\@itemdepth>\thr@@
    \@toodeep
  \else
    \advance\@itemdepth\@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
  \fi
  \@ifnextchar[{\@itemlabel@{\@compactitem@}}{\@compactitem@}}
```

The width of the label is adjusted before starting the list if necessary.

```
\def\@compactitem@{%
  \@adjust@item@labelwidth
  \expandafter\list\csname\@itemitem\endcsname{%
    \parsep\plparsep
    \itemsep\plitemsep
    \topsep\pltopsep
    \partopsep\plpartopsep
    \labelwidth
      \csname leftmargin\romannumeral\@listdepth\endcsname
    \advance\labelwidth-\labelsep
    \let\makelabel\itemlabel}}
\let\endcompactitem\endlist
```

`itemize` If the user has requested it, we redefine the `itemize` environment.

```
\if@plnewitem
  \def\itemize{%
    \ifnum \@itemdepth >\thr@@
      \@toodeep
    \else
      \advance\@itemdepth\@ne
      \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \fi
    \@ifnextchar[{\@itemlabel@{\@itemize@}}{\@itemize@}}
```

The width of the label is adjusted before starting the list if necessary.

```
  \def\@itemize@{%
    \@adjust@item@labelwidth
    \expandafter\list\csname\@itemitem\endcsname{%
      \labelwidth
        \csname leftmargin\romannumeral\@listdepth\endcsname
      \advance\labelwidth-\labelsep
      \let\makelabel\itemlabel}}
\fi
```

10.9 Descriptive Environments

`asparadesc` A bit easier than `asparaenum` since the label handling is easier.

```
\def\asparadesc{%
  \list{}{%
    \labelwidth\z@
    \labelsep.5em
    \leftmargin\z@
    \parsep\parskip
    \itemsep\z@
    \topsep\z@
    \partopsep\parskip
    \itemindent\parindent
    \advance\itemindent\labelsep
    \let\makelabel\paradescriptionlabel}}
\let\endasparadesc\endlist
```

`inparadesc` Again a bit easier than `inparaenum` since the label handling is easier.

```
\def\inparadesc{%
  \let\@itemlabel\@empty
  \let\@item\pl@item
  \let\makelabel\paradescriptionlabel
  \ignorespaces}
\let\endinparadesc\ignorespacesafterend
```

`compactdesc` The `compactdesc` environment is copied from `article.cls` with the skips set to the `\pl...` skips.

```
\def\compactdesc{%
  \list{}{%
    \parsep\plparsep
    \itemsep\plitemsep
    \topsep\pltopsep
    \partopsep\plpartopsep
    \labelwidth\z@
    \itemindent-\leftmargin
    \let\makelabel\descriptionlabel}}
\let\endcompactdesc\endlist
```

10.10 Blank Environments

`asparablank` This list formats its entries as if there was no list. Thus no test for the list depth is performed. To avoid unnecessary white space, `\labelsep` is set to zero. If the optional argument of `\item` is used, some separating space has to inserted “by hand”.

```
\if@pldefblank
  \def\asparablank{%
    \list{}{%
      \labelwidth\z@
      \labelsep\z@
      \leftmargin\z@
      \parsep\parskip
      \itemsep\z@
      \topsep\z@
      \partopsep\parskip
      \itemindent\parindent
      \advance\itemindent\labelsep
      \def\makelabel##1{##1}}}
  \let\endasparablank\endlist
```

`inparablank` The `inparablank` environment handles the "white space problem" automatically.

```
  \def\inparablank{%
    \let\@itemlabel\@empty
    \let\@item\pl@item
    \ignorespaces}
  \let\endinparablank\ignorespacesafterend
\fi
```

10.11 Setting Defaults

Here are the macros to define the default labels and left margins for itemized and enumerated environments.

`\setdefaultitem` The macro for the item labels just redefines `\labelitem`n.

```
\def\setdefaultitem#1#2#3#4{%
  \if@empty{#1}{}{\def\labelitemi{#1}}%
  \if@empty{#2}{}{\def\labelitemii{#2}}%
  \if@empty{#3}{}{\def\labelitemiii{#3}}%
  \if@empty{#4}{}{\def\labelitemiv{#4}}}
```

`\defaultitem` This macro is deprecated and only kept for backward compatibility.

```
\def\defaultitem{%
  \PackageWarning{Paralist}{Macro \protect\defaultitem\space
    deprecated. \MessageBreak Use macro \protect\setdefaultitem
    \space instead!! \MessageBreak Used}
  \setdefaultitem}
```

`\setdefaultenum` The macro to define the enumerated labels uses the same mechanism as the optional argument of the enumerated environments. First, we set `\@enumctr` to `enum`n, then `\@enumlabel@` is used to format the counter. As explained above we have to add the extra bracket `[` to avoid losing braces `{}` around the whole argument. Note that the state of `\if@alwaysadjust` has to be saved and restored.

```
\newif\if@pltemp
\def\setdefaultenum#1#2#3#4{%
  \if@plneveradjust\@pltemptrue\else\@pltempfalse\fi
  \@plneveradjusttrue
  \if@empty{#1}{}{%
    \def\@enumctr{enumi}%
    \@enumlabel@{\relax}[[#1]]%
  \if@empty{#2}{}{%
    \def\@enumctr{enumii}%
    \@enumlabel@{\relax}[[#2]]%
  \if@empty{#3}{}{%
    \def\@enumctr{enumiii}%
    \@enumlabel@{\relax}[[#3]]%
  \if@empty{#4}{}{%
    \def\@enumctr{enumiv}%
    \@enumlabel@{\relax}[[#4]]%
  \if@pltemp\@plneveradjusttrue\else\@plneveradjustfalse\fi
  \@plmylabelfalse
  \relax}
```

\defaultenum This macro is deprecated and only kept for backward compatibility.

```
\def\defaultenum{%
  \PackageWarning{Paralist}{Macro \protect\defaultenum\space
    deprecated. \MessageBreak Use macro \protect\setdefaultenum
    \space instead!! \MessageBreak Used}
  \setdefaultitem}
```

\setdefaultleftmargin Finally a macro to define all the left margins for the lists.

```
\def\setdefaultleftmargin#1#2#3#4#5#6{%
  \if@empty{#1}{}{\leftmargini#1}%
  \if@empty{#2}{}{\leftmarginii#2}%
  \if@empty{#3}{}{\leftmarginiii#3}%
  \if@empty{#4}{}{\leftmarginiv#4}%
  \if@empty{#5}{}{\leftmarginv#5}%
  \if@empty{#6}{}{\leftmarginvi#6}%
  \relax}
```

\defaultleftmargin This macro is deprecated and only kept for backward compatibility.

```
\def\defaultleftmargin#1#2#3#4{%
  \setdefaultleftmargin{#1}{#2}{#3}{#4}{}{}%
  \PackageWarning{Paralist}{Macro \protect\defaultleftmargin
    \space deprecated. \MessageBreak Use macro
    \protect\setdefaultleftmargin\space instead!!
    \MessageBreak But note that it has six arguments!
    \MessageBreak Used}}
```

10.12 Points or no Points?

\pl@pointxxxenum We define macros for the options pointlessenum and pointedenum. First the common part consisting of the format of the counters itself and the references. The references to an item of level N is generated by using `\p@enumN\theenumN`, so we have to make all `\p@enumN`'s do nothing.

```
\def\pl@pointxxxenum{%
  \def\theenumi{\arabic{enumi}}%
  \def\theenumii{\theenumi.\arabic{enumii}}%
  \def\theenumiii{\theenumii.\arabic{enumiii}}%
  \def\theenumiv{\theenumiii.\arabic{enumiv}}%
  \def\p@enumi{}%
  \def\p@enumii{}%
  \def\p@enumiii{}%
  \def\p@enumiv{}}
```

`\pl@pointedenum` `\pl@pointlessenum`

The format of the labels either adds a point or it doesn't.

```
\def\pl@pointedenum{%
  \def\labelenumi{\theenumi.}%
  \def\labelenumii{\theenumii.}%
  \def\labelenumiii{\theenumiii.}%
  \def\labelenumiv{\theenumiv.}}
\def\pl@pointlessenum{%
  \def\labelenumi{\theenumi}%
  \def\labelenumii{\theenumii}%
  \def\labelenumiii{\theenumiii}%
  \def\labelenumiv{\theenumiv}}
```

`\pointedenum` `\pointlessenum`

Now the user level macros to turn the new feature on.

```
\def\pointedenum{\pl@pointxxxenum\pl@pointedenum}
\def\pointlessenum{\pl@pointxxxenum\pl@pointlessenum}
```

The options just use those user level macros globally.

```
\if@plpointedenum\pointedenum\fi
\if@plpointlessenum\pointlessenum\fi
```

10.13 Configuration File

Read the file `paralist.cfg` if it exists and the `nocfg` was not given.

```
\if@plloadcfg
  \InputIfFileExists{paralist.cfg}{%
    \PackageInfo{Paralist}{%
      Using the configuration file paralist.cfg}}{}
\fi
⟨/package⟩
```